

⑨ 日本国特許庁 (JP)

⑩ 特許出願公開

⑪ 公開特許公報 (A) 平2-151930

⑫ Int. Cl. 5

G 06 F 9/38
12/00

識別記号

3 5 0 X
3 0 3 G

府内整理番号

7361-5B
8841-5B

⑬ 公開 平成2年(1990)6月11日

審査請求 未請求 請求項の数 2 (全11頁)

⑭ 発明の名称 ストアバッファ管理方式

⑮ 特 願 昭63-306137

⑯ 出 願 昭63(1988)12月5日

⑰ 発明者 宮沢 文彦 東京都港区芝5丁目33番1号 日本電気株式会社内

⑱ 出願人 日本電気株式会社 東京都港区芝5丁目33番1号

⑲ 代理人 弁理士 山川 政樹 外2名

明細書

1. 発明の名称

ストアバッファ管理方式

2. 特許請求の範囲

(1) ストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算装置と主記憶装置とを含む情報処理装置において、前記ストアバッファに登録中であり前記主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のストアを該ロード命令の処理に優先して前記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、該ストア処理完了まで前記ロード命令の待合わせを行う手段と、前記ロード命令と同一主記憶アドレスへのストア処理が終了したならば前記ロード命令の待合わせの解除を行う手段とを含むことを特徴とするストアバッファ管理方式。

(2) ストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算処理と主記憶

装置とを含む情報処理装置において、ストアバッファに登録中であり主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行の該ストア命令のみを前記ロード命令の処理に優先して前記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、該ストア処理完了まで前記ロード命令の待合わせを行う手段と、前記ロード命令と同一主記憶アドレスへのストア処理を終了したならば前記ロード命令抑止の解除を行う手段と、すでに主記憶装置に書き込んでしまつたストアバッファ内実行済みの該ストア命令に対しては再度主記憶装置に対して書き込むことがないように制御する手段とを含むことを特徴とするストアバッファ管理方式。

3. 発明の詳細な説明

〔産業上の利用分野〕

本発明はストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算装置と主記憶装置とを含む情報処理装置における命令の

高速化処理に係り、特にストア命令の高速処理を実現するためのストアバッファ管理方式に関するものである。

〔従来の技術〕

従来のパイプライン型情報処理装置においては、ロード命令処理が優先的に主記憶との間で処理を行ないストア命令処理はストアバッファに格納された後で、ロード命令が主記憶とアクセスを行なつていなければ、主記憶に対する書き込み処理を行なう。そして、先行するストア命令と同一主記憶アドレスに対して後続ロード命令が存在した場合、ストア命令処理をロード命令処理に優先して処理しなければ、主記憶のデータに誤まりが生じてしまう。

そこで、主記憶の同一アドレスへの先行ストア命令に対し後続ロード命令が存在する場合には、ストアバッファ内の未実行のストア命令処理をそのロード命令処理よりも優先して処理を行ない、ストアバッファ内の全ての未実行のストア処理が終了したならば、そのロード命令処理を実行する

ようになつていた。

〔発明が解決しようとする課題〕

上述した従来のストアバッファ管理方式では、ストアバッファに登録中であり未実行であるストアアドレスに対し後続するロード要求が同一アドレスに対し実行しようとした場合、ストアバッファ内に登録されている個々のストアアドレスに対し、どのストアアドレスがそのロード要求と一致しているのかを示す手段を持つていなかつたために先行ストアアドレスと後続ロードアドレスが一致した場合にはストアバッファ内のすべての主記憶に対して未実行のストア処理をそのロード要求に優先して処理しなければならず、そのロード要求の処理を開始する迄に時間を費やすという課題があつた。

〔課題を解決するための手段〕

本発明のストアバッファ管理方式は、ストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算装置と主記憶装置とを含む情報処理装置において、上記ストアバッファに登

- 4 -

録中であり、上記主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のストアをそのロード命令の処理に優先して上記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、そのストア処理完了まで上記ロード命令の待合せを行う手段と、上記ロード命令と同一主記憶アドレスへのストア処理が終了したならば上記ロード命令の待合せの解除を行う手段とを含むものである。

また、本発明の別の発明によるストアバッファ管理方式は、上記の情報処理装置において、ストアバッファに登録中であり主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のそのストア命令のみを上記ロード命令の処理に優先して上記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、そのストア処理完了まで上記ロード命令の待合せを行う手段と、上記ロード命令と

同一主記憶アドレスへのストア処理を終了したならば上記ロード命令抑止の解除を行い手段と、ナビゲーション記憶装置に書き込んでしまつたストアバッファ内実行済みのそのストア命令に対しては再度主記憶装置に対して書き込むことがないように制御する手段とを含むものである。

〔作用〕

本発明においては、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、ストアバッファ内の同一主記憶アドレスに対してのストア処理が全て終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換える。また、別の発明においては、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、ストアバッファ内の同一主記憶アドレスに対してのストア処理のみを実行し処理が終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換える。

- 5 -

-238-

- 6 -

〔実施例〕

以下、図面に基づき本発明の実施例を詳細に説明する。

第1図は本発明の一実施例を示すブロック図である。

図において、1はストアバッファアドレス書込みレジスタ、2, 3, 4はストアアドレス登録レジスタ、5, 6, 7は一致検出コンパレーター、8は主記憶書込み待ちアドレスレジスタ、9は拂出し制御部、10はストアバッファ読み出しレジスタ、11はインクリメンタ、12はストアバッファデータ書込みレジスタ、13はインクリメンタ、14はストアデータバッファ、15は主記憶書込み待ちデータレジスタ、16はインクリメンタである。

そして、ストアバッファに登録中であり、主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のストアをそのロード命令の処理に優先してロード命

- 7 -

第4図は本発明の一実施例のタイムチャートで、(a)はサイクルを示したものであり、(b)はステージIF、(c)はステージAC、(d)はステージAT、(e)はステージCA、(f)はステージEX、(g)はステージST、(h)はストアアドレス登録レジスタSA、(i)は主記憶、(j)はライトアドレスポインタ、(k)はライトデータポインタ、(l)はリードデータポインタを示したものである。

第5図はバイブライン処理の一例を示す図である。

つぎに第1図に示す実施例の動作を第2図ないし第5図を参照して説明する。

バイブライン処理の一例を示す第5図においては、ステージIFからステージSTまで6ステージに分割されている。そして、ステージIFは命令を取出す処理、ステージACはオペランドを生成する処理、ステージATは論理アドレスを実アドレスに変換する処理、ステージCAはオペランドを読み出す処理、ステージEXは読み出されたオペラドの演算処理、ステージSTは演算結果の格納処

理のアクセスアドレスと同一主記憶アドレスに対して実行するよう構成されている。また、一致検出コンパレーター5～7と拂出し制御部9はストア処理完了までロード命令の待合を実行する手段を構成し、拂出し制御部9はロード命令と同一主記憶アドレスへのストア処理が終了したならばロード命令の待合を解除する手段を構成している。

第2図は第1図における拂出し制御部9の詳細を示す図である。

この第2図において、17はフラグバッファレジスタ、18はアダーラジスタ、19はレジスタ、20はデクリメンタである。

第3図は通常ストア処理のタイムチャートで、(a)は「第一例」を示し、(b)は「第二例」を示す。

そして、(a)はサイクルを示したものであり、(b)はステージIF、(c)はステージAC、(d)はステージAT、(e)はステージCA、(f)はステージEX、(g)はステージST、(h)はストアアドレス登録レジスタSA、(i)は主記憶を示したものである。

- 8 -

理を行う。

以降本発明の実施例をこの第5図に示すバイブルайнステージを基にして説明する。

まず、通常ストア処理について第3図に示すタイムチャートを参照して説明する。(a)に示す「第一例」として時間t₁においてストアA命令処理が開始されると、時間t₂においてストアAに関するストアアドレスが第1図のストアアドレス登録レジスタ2～4に登録される。そして、時間t₃において演算が行なわれ、時間t₄においてストアすべきデータがステージSTの第1図に示すストアデータバッファ14に登録される。このように、ストア命令においてはアドレスがストアバッファに登録されるタイミングとストアデータがストアバッファに登録するタイミングが異なつている。そして、時間t₅においてアドレスBに対してのロードB命令がステージATに存在したとすると、本発明における情報処理装置においてはロード命令処理がストア命令処理よりも優先して処理されるように制御されているため、時間t₆に

- 9 -

-239-

-10-

においてロードB命令が主記憶とのアクセスを行なない、ストアA命令はストアバッファで処理の待合をせを行ない、時間t₁においてストアA命令の主記憶とのアクセスが行なわれる。

そして、(b)に示す「第二例」として時間t₁においてストアC命令の処理が始まり、時間t₂においてロードC命令の処理が始まるとすると、時間t₂において先行するストアC命令が完了しないうちに後続ロードC命令を実行しようとする。ここで、後続ロードC命令が先に処理されてしまうならば主記憶から間違つたデータを取出してしまう。そこで、先行するストアC命令を優先する必要があり、時間t₂においてストアC命令の主記憶とのアクセスが行なわれ、時間t₃においてロードC命令の主記憶とのアクセスが行なわれる。

本発明は第3図の(b)に示す「第二例」における制御方法であり、第1図に示す実施例および第4図に示すタイムチャートにより具体的に説明する。

第4図のタイムチャートにおいて、ストアD、

ストアE、ストアE、ストアF、ロードG、ロードH、ロードEの順に時間t₁～t₇において各命令がステージIFに受けられるとすると、ストア命令は、

IF → AC → AT → CA → EX → ST

と処理が進み、ロード命令は

IF → AC → AT

と処理が進む。そして、ストア命令においては、AT → CA のタイミングで各ストア命令のアドレスがステージ8Aに登録される(t₁～t₂)。

EX → ST のタイミングでストア命令のストアデータが第1図のストアデータバッファ14に登録される(t₂～t₃)。そして、ストアアドレスとストアデータが描つたストア命令はストアバッファ→主記憶への掲出しタイミングを待つ。この掲出しタイミングは「ロード命令が主記憶を使用しない時」である。よつて、ロード命令がステージAT→主記憶へ掲出するタイミング以外のときにストアバッファから主記憶への掲出しが可能となる。

そして、時間t₃ではストアDが主記憶に掲出

-12-

され、時間t₄ではロードGの主記憶アクセスが行なわれ、時間t₅ではロードHの主記憶アクセスが行なわれる。

時間t₁のステージATにアドレスEに対するロードE命令が存在する。また、ストアバッファ内にはアドレスEに対するストアE命令が存在する。このとき、「第二例」で説明したように、ストアバッファ内から未実行のストアE命令が終了するまでロードE命令の処理よりもストア命令処理を優先させる。よつて、時間t₂においてストアEが主記憶に掲出され、時間t₃において次のストアEが主記憶に掲出される。この時点でストアバッファ内には未実行のストアE命令はなくなるので、第1図に示す掲出し制御部9においてストアバッファの掲出し抑止を行い、時間t₃においてロードE命令の主記憶アクセスが行なわれ、時間t₄においてストアFの主記憶掲出しが行なわれる。

そして、ストア命令の第1図に示すストアアドレス登録レジスタ2～4への書き込みはストアバッ

ファアドレス書き込みレジスタ1によつて、ストアデータバッファ14への書き込みはストアバッファデータ書き込みレジスタ12によつてそれぞれ行なわれ、ストアバッファから主記憶への掲出しへストアバッファ読出しレジスタ10によつて行なわれる。

掲出し制御部9においては、ストアバッファ内に未実行の、「ロード命令よりも優先しなければならないストア命令」がいくつ存在するかが把握されている。この掲出し制御部9の詳細を第2図に示す。第1図において各ストアアドレス登録レジスタ2～4の値と後続アドレスの値が一致検出コンバレータ5～7によつて一致が調べられ、もし一致しており、後続命令がロード命令であるならば第2図のフラグバッファレジスタ17にフラグがセットされる。また、一致検出コンバレータ5～7のTotal1数、つまり、掲出されなければならないストア命令数がレジスタ19に示される。そして、第1図のストアバッファ読出しレジスタ10によつてストアバッファの掲出しが行なわれ、

-13-

-240-

-14-

第2図のフラグバッファレジスタ17からの出力が「1」であるならば1カウンタであるデクリメント20によりレジスタ19の減算を行い、このレジスタ19が「0」になつたならばストアバッファ内に掲出し必要なストア命令は無いとしてストアバッファからの掲出しを抑制し、ロード命令を開始するように制御の変更を行なう。

第6図は本発明の他の実施例を示すブロック図である。

この第6図において第1図と同一符号のものは相当部分を示し、21は掲出し制御部である。

そして、ストアバッファに登録中であり、主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のそのストア命令のみをロード命令の処理に優先してロード命令のアクセスアドレスと同一主記憶アドレスに対して実行するように構成されている。また、一致検出コンバレーター5～7と掲出し制御部21はストア処理完了までロード命令の待合わ

せを行なう手段を構成し、掲出し制御部21はロード命令と同一主記憶アドレスへのストア処理を終了したならばロード命令抑止の解除を行なう手段を構成すると共に、すでに主記憶装置に書込んでしまつたストアバッファ内実行済みのそのストア命令に対しては再度主記憶装置に対して書込むことがないように制御する手段を構成している。

第7図は第6図における掲出し制御部21の詳細を示す図である。

この第7図において22はデコーダ、23は有効ストア指示フラグバッファ、24はエンコーダ、25は優先ストア指示フラグバッファ、26はエンコーダ、27はストアバッファ脱出し指定レジスタ28はセレクタ、29はストアバッファ脱出レジスタ、30はアダーダー、31はレジスタ、32はデクリメンタ、33₁、33₂…33_nは論理積回路、34は優先ストア指示信号線である。

第8図は本発明の他の実施例のタイムチャートで、(a)はサイクルを示したものであり、(b)はステージIF、(c)はステージAC、(d)はステージAT、(e)はステージCA、

-15-

-16-

(a)はステージEX、(b)はステージST、(c)はストアアドレス登録レジスタSA、(d)は主記憶、(e)はライトアドレスポインタ、(f)はライトデータポインタ、(g)はリードポインタを示したものである。

つぎに第6図に示す実施例の動作を第7図および第8図を参照して説明する。なお、バイブライン処理については前述の第1図に示す実施例と変わらないので、ここでの説明を省略する。

まず、本発明は第3図の(b)に示す「第二例」における制御方法である。

第8図に示すタイムチャートにおいて、ストアD、ストアE、ストアF、ストアG、ロードG、ロードH、ロードE、ストアGの順に時間t₁からt₅において各命令がステージIFに受けられるとすると、ストア命令は

IF → AC → AT → CA → EX → ST
と処理が進み、ロード命令は
IF → AC → AT
と処理が進む。そして、ストア命令においては
AT → CAのタイミングで各ストア命令のアドレ

スが第6図のストアアドレス登録レジスタ2～4に登録される(t₁～t₂、t₃～t₄)。また、EX → STのタイミングでストア命令のストアデータが第6図のストアデータバッファ14に登録される(t₂～t₃、t₄～t₅)。

そして、ストアアドレスとストアデータが掛つたストア命令はストアバッファから主記憶への掲き出しタイミングを待つ。この掲き出しタイミングは「ロード命令が主記憶を使用しない時」である。よつて、ロード命令がステージATから主記憶へアクセスするタイミング以外のときにストアバッファから主記憶への掲き出しが可能となる。

時間t₁ではストアDが主記憶に掲き出され、時間t₂ではロードGの主記憶アクセスが行なわれ、時間t₃ではロードHの主記憶アクセスが行なわれる。時間t₄のステージATに主記憶アドレスEに対するロードE命令が存在する。このとき、ストアバッファ内には主記憶アドレスEに対するストアE命令が存在する。このとき第3図の(b)に示す「第二例」で述べたように、ストアバッ

-17-

-241-

-18-

FA内から主記憶に對して未実行のストアE命令が終了するまでロードE命令の処理よりもストアE命令処理を優先させる。

このとき、この第6図に示す実施例においては、ストアバッファ内に主記憶アドレスEに対するストア命令は2つ存在するため、この2つのストア命令Eの実行を優先的に行なう。そして、時間t₁₁において最初のストアE命令がストアバッファから主記憶に拂き出され、時間t₁₂において2番目のストアE命令が主記憶に拂き出される。この時点でストアバッファ内には主記憶に對して未実行のストアE命令はなくなるので、拂出し制御部21においてストアバッファの拂き出し抑止を行ない、時間t₁₃においてロードE命令の主記憶アクセスが行なわれ、時間t₁₄においてストアバッファ内の主記憶に對して未実行のストアF命令が行なわれ、時間t₁₅においてはストアG命令が主記憶アドレスGに對して実行される。

そして、ストア命令のストアアドレス登録レジスタ2～4への書込みは、ストアバッファアドレ

ス書込みレジスタ1によつて、ストアデータバッファ14への書込みはストアバッファデータ書込みレジスタ12によつてそれぞれ行なわれ、ストアバッファから主記憶への拂き出しが第7図のストアバッファ読み出しレジスタ29によつて行なわれる。

拂出し制御部21においては、ストアバッファ内に未実行の「ロード命令よりも優先しなければならないストア命令」がいくつ存在するか把握しており、各ワードに対し拂き出さなくてはならない命令であるかをフラグを立てることによつて示す。このフラグが立つているストア命令についてのみストアバッファからの拂き出しを行うように拂出し制御部21で制御する。また、そのストア処理終了後、ストアバッファ内の主記憶に對して未実行であるストア命令のワード位置にストアバッファ読み出しレジスタの値を戻しておく必要があり、すでに主記憶に對して拂き出してしまつたストア命令に對しては再度ストアすることができないよう拂出し制御部21で制御を行う。

-20-

第7図にこの拂出し制御部21の実施例を示す。

通常ストア処理時は有効ストア指示フラグバッファ23を参照する。そして、この有効ストア指示フラグバッファ23はストアアドレス書込み時に指定ワードにフラグ「1」を立て、ストアバッファ拂き出し時にリセットを行なうフラグバッファで、各ワードのストア命令が主記憶に對して実行済であるか、未実行であるかを示す。

また、ストアバッファ読み出し指定レジスタバッファ27には0～Nまでストアバッファのワード数の値がそれぞれにセットされている。そして、有効ストア指示フラグバッファ23からの出力信号をエンコーダ24によつてストアバッファ内未実行のストア命令のワード位置を指定するようエンコードを行ないセレクタ28によりストアバッファ読み出し指定レジスタバッファ27からの出力信号のうちストアバッファから拂き出すべきワード位置を指定してストアバッファ読み出しレジスタ29のセットを行う。

そして、前述の第3図の(2)の「第二例」に示す

ような同一記憶アドレスに對して未実行のストア命令とロード命令が競合する場合には、論理積回路33₁～33₄において第6図に示す一致検出コンペレーター5～7の値が「1」であり（先行ストアアドレス=後続ロードアドレス）かつロードリクエスト処理要求時であり、さらに主記憶に對して未実行のストアリクエスト（有効ストア指示フラグバッファ23の出力）である場合、優先ストア指示フラグバッファ25の対応ワードにフラグを立てる。この優先ストア指示フラグバッファ25の出力信号をエンコーダ26によつてデコードを行ない、このエンコーダ26によりどのストア命令処理を行なうかの判断を行ないセレクタ28を通してストアバッファ読み出し指定レジスタバッファ27からの出力信号を選択しストアバッファ読み出しレジスタ29のセットを行う。そして、アダ-30においては優先処理すべきストア命令がストアバッファ内にいくつ存在するかを計算しており、レジスタ31にセットする。また、優先ストア処理が1つ実行される度にデクリメント32に

よつてレジスタ31の更新を行なう。そして、レジスタ31が「0」である場合は通常ストア処理時であり、レジスタ31が「1」である場合には同一主記憶アドレスへのアクセスに関しストアバッファ内未実行ストア命令とロード命令が競合する場合であり、セレクタ28の選択信号に使われる。

有効ストア指示フラグバッファ23のリセットはストアバッファ読出しレジスタ28にセットされようとするワード位置のフラグに対し行なわれる。そして、優先ストア指示フラグバッファ25から一度読出されたワード位置のフラグは同一タイミングでリセットが行なわれる。つまり、有効ストア指示フラグバッファ23内および優先ストア指示フラグバッファ25内には主記憶に対して未実行のストア命令のワード位置のフラグのみが「1」となつている。

〔発明の効果〕

以上説明したように本発明は、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、

-23-

図はパイプライン処理の一例を示す図、第6図は本発明の他の実施例を示すブロック図、第7図は第6図における掲出し制御部の詳細を示す図、第8図は本発明の他の実施例のタイムチャートである。

1.....ストアバッファアドレス書き込みレジスタ、2～4.....ストアアドレス登録レジスタ、5～7.....一致検出コンバレーター、8.....主記憶書き込み待ちアドレスレジスタ、9.....掲出し制御部、10.....ストアバッファ読出しレジスタ、11.....インクリメンタ、12.....ストアバッファデータ書き込みレジスタ、13.....インクリメンタ、14.....ストアデータバッファ、15.....主記憶書き込み待ちデータレジスタ、16.....インクリメンタ、17.....フラグバッファレジスタ、18.....アダル、19.....レジスタ、20.....デクリメンタ、21.....掲出し制御部、22.....デコーダ、23.....有効ストア指示フラグバッファ、24.....エンコーダ

ストアバッファ内の同一主記憶アドレスに対してのストア処理が全て終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換えることにより、不必要的ストア処理によりロード命令の主記憶とのアクセスが遅れることがなくなるという効果がある。

また、本発明は、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、ストアバッファ内の同一主記憶アドレスに対してのストア処理のみを実行し処理が終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換えることにより、不必要的ストア処理によりロード命令の主記憶とのアクセスが遅れることがなくなるという効果がある。

4. 図面の簡単な説明

第1図は本発明の一実施例を示すブロック図、第2図は第1図における掲出し制御部の詳細を示す図、第3図は通常ストア処理のタイムチャート、第4図は本発明の実施例のタイムチャート、第5

-24-

ダ、25.....優先ストア指示フラグバッファ、26.....エンコーダ、27.....ストアバッファ読出し指定レジスタ、28.....セレクタ、29.....ストアバッファ読出しレジスタ、30.....アダル、31.....レジスタ、32.....デクリメンタ、33～33a.....論理積回路。

特許出願人 日本電気株式会社

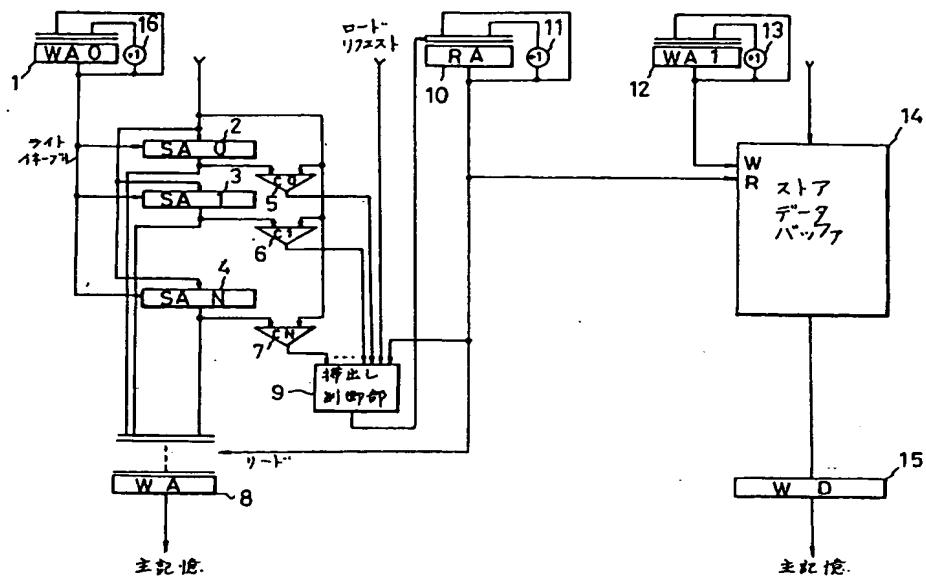
代理人 山川政樹(ほか2名)

-25-

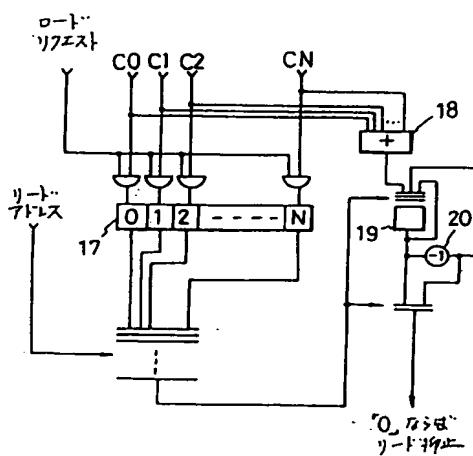
-243-

-26-

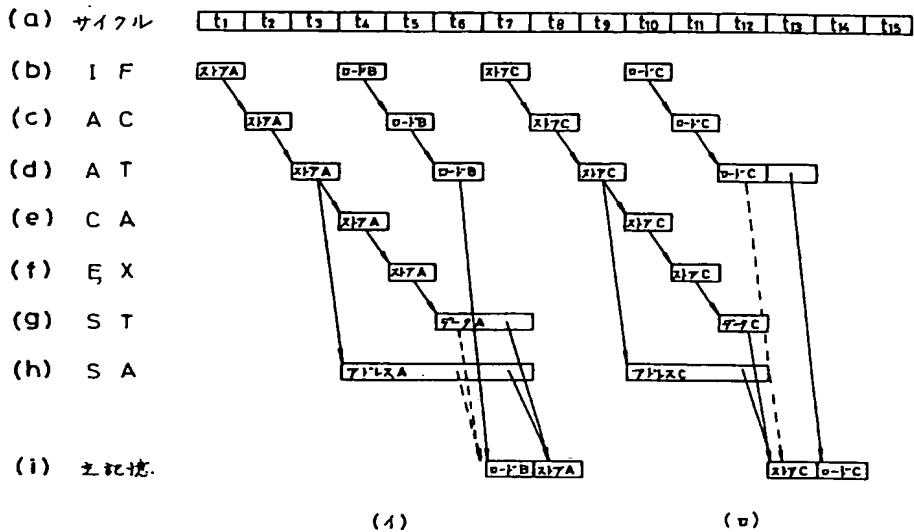
第 1 圖



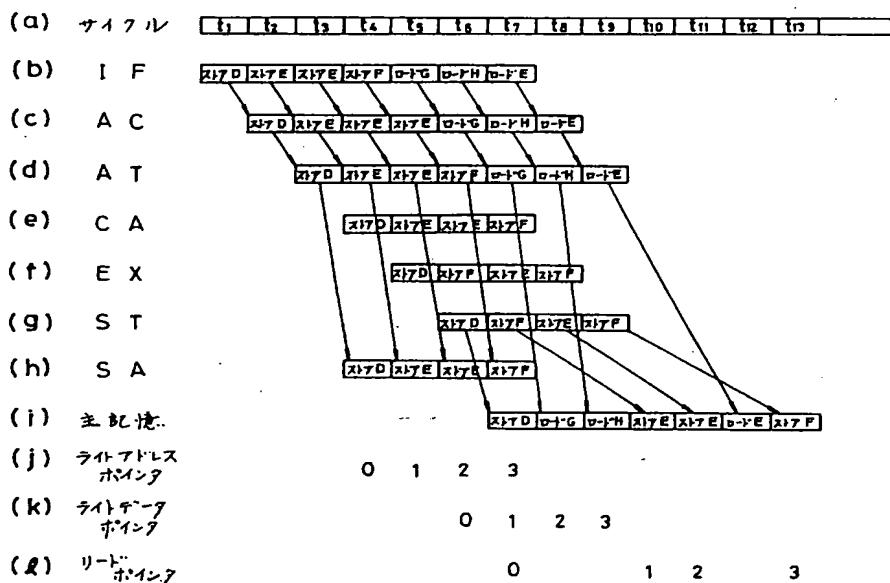
第 2 章



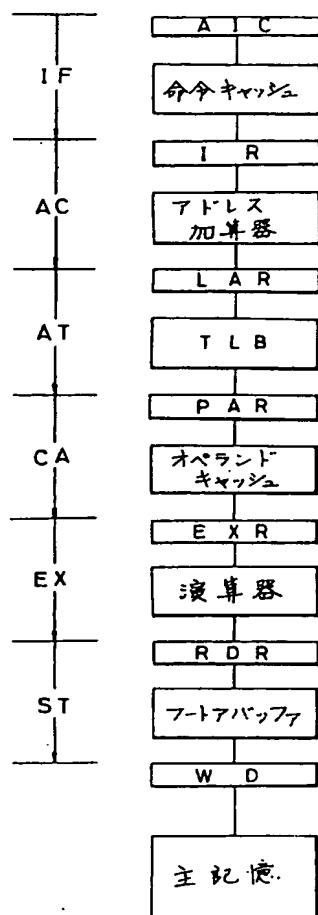
第 3 図



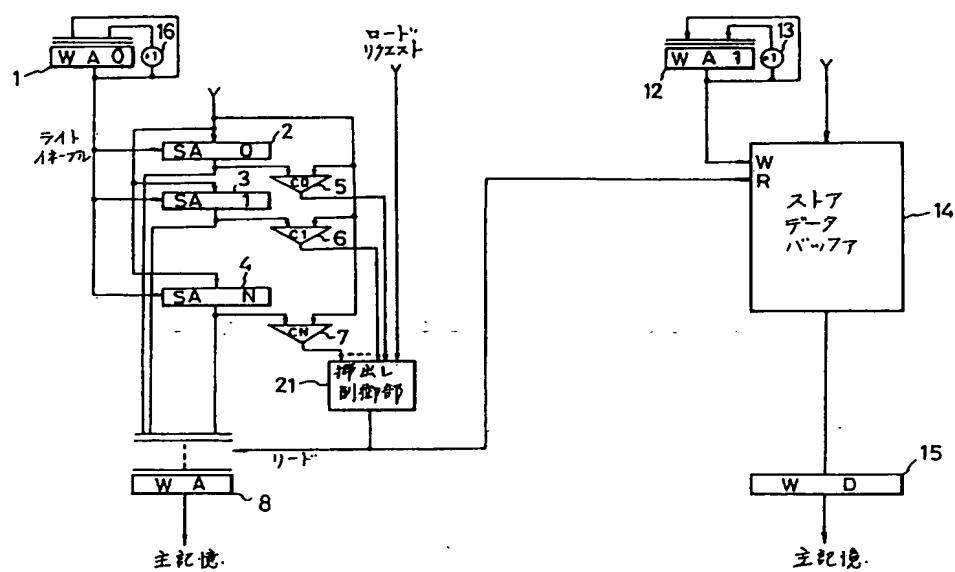
第 4 図



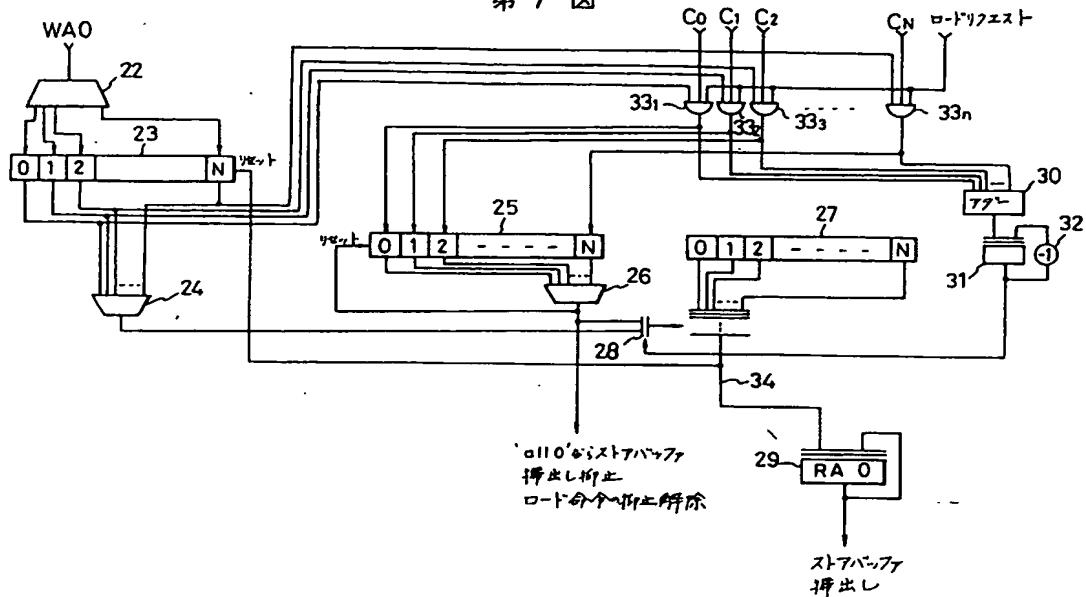
第 5 図



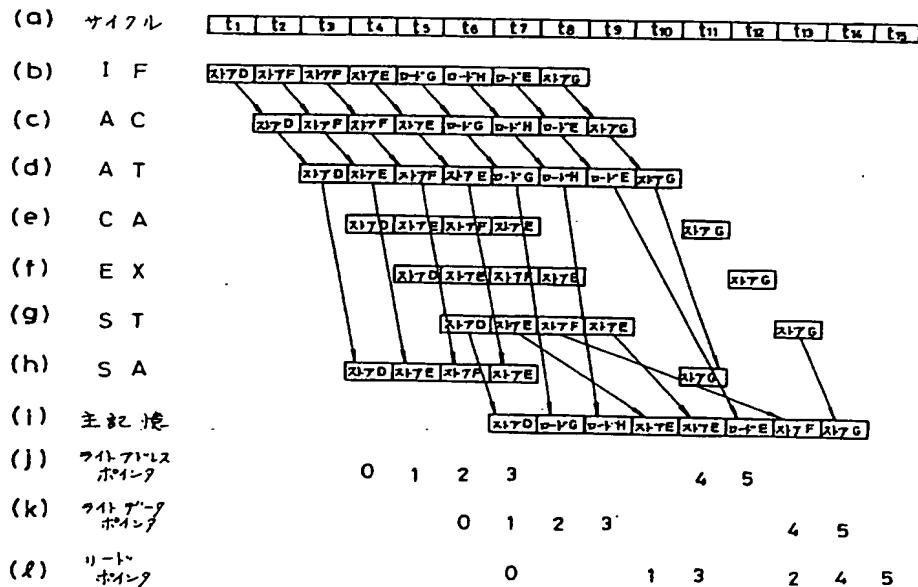
第 6 回



第 7 図



第 8 図



JAPANESE LAID-OPEN PATENT APPLICATION

H02-151930 (1990)

(19) Japan Patent Office (JP)

(11) Publication No. H02-151930

(12) Published Unexamined Patent Application (A) (43) Publication Date June 11, 1990

(51) Int. Cl.⁵ Ident. Code In-House Ref. No.

G 06 F 9/38	350	X	7361-5B
12/00	303	G	8841-5B

No examination request Number of Claims 2 (totally 11 pages)

(54) Title of the Invention

STORAGE BUFFER MANAGEMENT SYSTEM

(21) Application No.

PA S63-306137

(22) Date of Filing

December 5, 1988 (Showa 63)

(72) Inventor

Fumihiko Miyazawa
C/O Nihon Electric Corporation
5-33-1 Shiba
Minato-ku, Tokyo

(71) Applicant

Nihon Electric Corporation
5-33-1 Shiba
Minato-ku, Tokyo

(74) Agent

Masaki Yamakawa, Attorney
(two others)

Specification

1. Title of the Invention

Storage buffer Management System

2. Claims

Claim 1

A storage buffer management system for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, and which comprises:

a means for queuing a load instruction until the storage management is completed and executes the unexecuted storage registered in a storage buffer in relation to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer, and

a means for removing the queue of the load instruction if the storage processing is completed to the same main memory address as the load instruction.

Claim 2

A storage buffer management system for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, and which comprises:

a means for queuing a load instruction until the storage management is completed and executes only the unexecuted storage instruction registered in a storage buffer in relation

to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer; and

a means for removing the load instruction inhibitor if the storage processing is completed to the same main memory address as the load instruction; and

a means for controlling so as to prevent an executed storage instruction in the storage buffer already written to the main memory device from being written again to the main memory device.

3. Detailed Description of the Invention

Industrial Applications

The present invention for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management relates to the speeding up of processing instructions, and especially relates to a storage buffer management method for realizing high speed processing of storage instructions.

Prior Art

Regarding conventional pipeline type information processing devices, writing processing to the main memory is performed while the load instruction has not yet accessed the main memory after the load instruction processing which has priority is processed between the main memory, and the storage instruction processing is stored in the storage buffer. Further, when there is a successive load instruction in relation to the same main memory address as a preceding storage instruction, an error can occur in the data of the main

memory if the storage instruction processing is not processed having priority over the load instruction processing.

Therefore, when there is a successive load instruction in relation to a preceding storage instruction of the same address of the main memory, the unexecuted storage instruction processing within the storage buffer is processed having priority over such load instruction processing, and is constructed such that such load instruction processing is executed after all unexecuted storage processing within the storage buffer has been completed.

Problems Overcome by the Invention

With the conventional storage buffer management system described above, when a successive load request is attempted to be executed to the same address in relation to an unexecuted storage address during registration to the storage buffer, since there is no means to indicate which storage address matches such a load address in relation to each individual storage address registered in the storage buffer, there is the problem that when a preceding storage address matches a successive load address, unexecuted storage processing in relation to all of the main memory within the storage buffer must be processed as having priority over the load request; otherwise the time until initiating the processing for such load request take too long.

Problem Resolution Means

The storage buffer management system of the present invention for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, and comprises a means for queuing a load instruction until the storage management is completed and executes the unexecuted storage registered in a storage buffer in relation to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a

successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer, and

a means for removing the queue of the load instruction if the storage processing is completed to the same main memory address as the load instruction.

Further, the storage buffer management system according to a separate invention of the present invention for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, comprises:

a means for queuing a load instruction until the storage management is completed and executes only the unexecuted storage instruction registered in a storage buffer in relation to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer, and

a means for removing the load instruction inhibitor if the storage processing is completed to the same main memory address as the load instruction, and

a means for controlling so as to prevent executed storage instructions in the storage buffer already written to the main memory device from being written again to the main memory device.

Operation

According to the present invention, when there is a successive load instruction in relation to the storage instruction within an unexecuted storage buffer that precedes the same main memory address, storage processing which has priority over the load processing is switched to load processing priority at the time of completion of all the storage processing in relation to the same memory address within the storage buffer. Further, in

another invention, when there is a successive load instructions in relation to the storage instruction within an unexecuted storage buffer that precedes the same main memory address, storage processing is only executed in relation to the same main memory address within the storage buffer, and the storage processing which has priority over the load processing is switched to load processing priority at the time of completion of the processing.

Embodiments

A detailed description of the Embodiment of the present invention is provided hereafter with reference to the drawings.

FIG 1 is a block diagram showing an example of the present invention.

In the drawing, numeral 1 indicates a storage buffer address writing register; numerals 2, 3, and 4 indicate storage address entry registers; numerals 5, 6, and 7 indicate match detection comparators; numeral 8 indicates a main memory write pending address register; numeral 9 indicates a sweeping controller; numeral 10 indicates a storage buffer read register; numeral 11 indicates an incrementer; numeral 12 indicates a storage buffer data writing register; numeral 13 indicates an incrementer; numeral 14 indicates a store data buffer; numeral 15 indicates a main memory write pending data register; and numeral 16 indicates an incrementer.

Further, when accessing a successive load instruction that coincides with an address of an unexecuted storage instruction in relation to the main memory device while registering in the storage buffer, a construction is provided so that an unexecuted storage registered in the storage buffer can be executed in relation to the same main memory address as the access address of the load instruction which has priority over the processing of such load instruction. Furthermore, the match detection comparators 5 through 7 and the sweeping controller 9 constitutes a means that queues a load instruction until the completion of the storage processing, and the sweeping controller 9 constitutes a means that releases the

queue of the load instruction at the time of the completion of the storage processing to the same main memory address as the load instruction.

FIG 2 illustrates the details of the sweeping controller 9 in FIG 1.

Referring to FIG 2, numeral 17 indicates a flag buffer register; numeral 18 indicates an adder; numeral 19 indicates a register; and numeral 20 indicates a decrementer.

FIG 3 is a time chart of normal storage processing where (X) indicates the “Primary Example” and (Y) indicates the “Secondary Example”.

Further, (a) shows a cycle; (b) is a stage IF; (c) is a stage AC; (d) is a stage AT; (e) is a stage CA; (f) is a stage EX; (g) is a stage ST; (h) is a store address entry register SA; and (i) shows a main memory.

FIG 4 is a time chart of an Embodiment of the present invention, and (a) shows a cycle; (b) is a stage IF; (c) is a stage AC; (d) is a stage AT; (e) is a stage CA; (f) is a stage EX; (g) is a stage ST; (h) is a store address entry register SA; (i) is a main memory; (j) is a write address pointer; (k) is a write data pointer; and (l) shows a read pointer.

FIG 5 is a diagram showing an example of pipeline processing.

The following describes the operation of the Embodiment shown in FIG 1 with reference to FIG 2 through FIG 5.

Referring to FIG 5 which shows an example of pipeline processing, the stages are divided into six stages from the stage IF through the stage ST. The stage IF is the process for fetching a instruction; the stage AC is the process for creating an operand; the stage AT is the process for converting a logical address to an actual address; the stage CA is the process for reading an operand; the stage EX is the process for calculating a read operand; and the stage ST is the process for storing calculation result.

An Embodiment of the present invention is described hereafter based on the pipeline stage shown in FIG 5.

First, a description in regards to the normal storage process will be given with reference to the time chart shown in FIG 3. When the storage A instruction process begins at the time of t_1 as the “Primary Example” shown in (X), the storage address for storage A at the time of t_4 is registered in the storage address entry registers 2 through 4 of FIG 1. Then, calculation is performed at the time of t_5 , and data that should be stored is registered in the storage data buffer 14 shown in FIG 1 of stage ST at the time of t_6 . In this way, the timing for registering an address into the storage buffer is different from the timing for registering storage data into the storage buffer regarding the storage instruction. Further, if there is a load B instruction in relation to the address B in stage AT at the time of t_6 , because the information processing device with the present invention is controlled so that the load instruction process can have processing priority over the storage instruction process, the load B instruction accesses the main memory at the time of t_7 , and the storage A instruction queues for processing at the storage buffer, and then, access with the main memory of the storage A instruction is performed at the time of t_8 .

Next, the process of the storage C instruction begins at the time of t_9 as the “Secondary Example” shown in (Y), and when the process of the load C instruction begins at the time of t_{10} , a successive load C instruction is executed before the preceding storage C instruction is completed at the time of t_{12} . Here, if the successive load C instruction is processed first, the wrong data may be fetched. Therefore, the preceding storage C instruction must take priority thereby allowing access the main memory of the storage C instruction at the time of t_{13} and accessing the main memory of the load C instruction at the time of t_{14} .

The present invention is a control method for the “Secondary Example” shown in (Y) of FIG 3, and a detailed description is given hereafter with reference to the Embodiment shown in FIG 1 and the time chart shown in FIG 4.

Referring to the time chart of FIG 4, when considering that each instruction is received in stage IF at the times of t_1 through t_7 in the order of storage D, storage E, storage E, storage F, load G, load H, and Load E, a storage instruction proceeds to
IF → AC → AT → CA → EX → ST, and
a load instruction proceeds to IF → AC → AT.

Regarding a storage instruction, an address of each storage instruction is registered in the stage SA ($t_4 \sim t_7$) with a timing of AT → CA. Storage data of a storage instruction is registered in the storage data buffer 14 of FIG 1 ($t_{[illeg.]} \sim t_{[illeg.]}$) with a timing of EX → ST. Then, a storage instruction where a storage address and storage data are gathered waits the timing for sweeping from the storage buffer → to the main memory. The timing of this sweeping is “the time when a load instruction does not use the main memory”. Accordingly, sweeping from the storage buffer to the main memory is possible every time except the timing when a load instruction is swept from the stage AT → to the main memory.

The storage D is swept to the main memory at the time of t_7 , and the main memory access of the load G is performed at the time of t_8 , and the main memory access of the load H is performed at the time of T_9 .

There is a load E instruction in relation to the address E in the stage AT at the time of t_9 . Also, there is the storage E instruction in relation to the address E within the storage buffer. At that time, as described in the “Secondary Example”, the storage instruction process takes priority over the process of the load E instruction until the unexecuted storage E instruction is completed within the storage buffer. Accordingly, the storage E is swept to the main memory at the time of t_{10} , and the subsequent storage E is swept to the main memory at the time of t_{11} . At this point, since there are no unexecuted storage E instructions within the storage buffer, the sweep of the storage buffer is inhibited in the sweeping controller 9 shown in FIG 1, and the main memory access of the load E instruction is performed at the time of t_{12} , and the main memory sweep of storage F is performed at the time of t_{13} .

Further, writing a storage instruction to the storage address entry registers 2 through 4 that are shown in FIG 1 is performed by the storage buffer address writing register 1, and writing to the storage data buffer 14 is performed by the storage buffer data writing register 12 respectively, and sweeping from the storage buffer to the main memory is performed by the storage buffer read register 10.

The sweeping controller 9 understands how many unexecuted “storage instructions that must take priority over a load instruction” exist within the storage buffer. A detailed description of this sweeping controller 9 is shown in FIG 2. Referring to FIG 1, the match detection comparators 5 through 7 examine whether or not a value of each of the storage address entry registers 2 through 4 matches with a successive address value. When it matches and the successive instruction is a load instruction, a flag shall be set in the flag buffer register 17 of FIG 2. Further, the number of Totn 1 of the match detection comparators 5 through 7, in other words, the number of storage instructions that must be swept, is shown in the register 19. Then, the storage buffer is swept by the storage buffer reading register 10 of FIG 1, and when the output from the flag buffer register 17 of FIG 2 is “1”, the calculation of the register 19 is performed by the decrementer 20 which is one counter, and when the register 19 becomes “0”, sweeping from the storage buffer is inhibited assuming there is no storage instruction that needs to be swept within the storage buffer, and then, the control change is performed so that a load instruction can be initiated.

FIG 6 is a block diagram showing another Embodiment of the present invention.

Referring to FIG 6, the same codes are equivalent to FIG 1, and numeral 21 is a sweeping controller.

Further, when accessing a successive load instruction that matches with an address of an unexecuted storage instruction in relation to the main memory device while registering in the storage buffer, a construction is provided so that only unexecuted storage instructions registered in the storage buffer can be executed in relation to the same main memory

address as the access address of the load instruction which has priority over the processing of such load instruction.

Furthermore, the match detection comparators 5 through 7 and the sweeping controller 21 constitutes a means that queues a load instruction until the completion of the storage processing, and the sweeping controller 21 constitutes a means that releases the queue of the load instruction at the time of the completion of the storage processing to the same main memory address as the load instruction, and also a means that controls the writing so that a storage instruction, which has been executed within the storage buffer and has been written in the main memory device, is not written again in the main memory device.

FIG 7 is a drawing showing the details of the sweeping controller 21 in FIG 6.

Referring to FIG 7, numeral 22 is a decoder; numeral 23 is a validity storage instruction flag buffer; numeral 24 is an encoder; numeral 25 is a priority storage instruction flag buffer; numeral 26 is an encoder; numeral 27 is a storage buffer read assignment register; numeral 28 is a selector; numeral 29 is a storage buffer read register; numeral 30 is an adder; numeral 31 is a register; numeral 32 is a decrementer; numerals 33₁, 33₂, ... 33_n are AND circuits; and numeral 34 is a priority storage instruction signal line.

FIG 8 is a time chart of another Embodiment of the present invention, (a) shows a cycle; (b) is a stage IF; (c) is a stage AC; (d) is a stage AT; (e) is a stage CA; (f) is a stage EX; (g) is a stage ST; (h) is a store address entry register SA; (i) is a main memory; (j) is a write address pointer; (k) is a write data pointer; and (l) shows a read pointer.

The operation of the Embodiment shown in FIG 6 is described hereafter, with reference to FIG 7 and FIG 8. In addition, there is no difference in comparison to the Embodiment shown in FIG 1 regarding the pipeline processing, therefore, the description will be omitted.

First, the present invention is a control method for the “Secondary Example” shown in (Y) of FIG 3.

Referring to the time chart of FIG 8, when considering that each instruction is received in stage IF at the times of t_1 through t_8 in the order of storage D, storage E, storage F, storage E, load G, load H, load E, and storage G, a storage instruction proceeds the processing to

$IF \rightarrow AC \rightarrow AT \rightarrow CA \rightarrow EX \rightarrow ST$, and

a load instruction proceeds to

$IF \rightarrow AC \rightarrow AT$.

Further, regarding a storage instruction, an address of each storage instruction is registered in the storage address entry registers 2 through 4 ($t_4 \sim t_7, t_{11}$) of FIG 6 with a timing of $AT \rightarrow CA$. Furthermore, storage data of a storage instruction is registered in the storage data buffer 14 of FIG 6 ($t_6 \sim t_9$, and t_{13}) with a timing of $EX \rightarrow ST$.

Then, a storage instruction where a storage address and storage data are gathered waits for the timing for sweeping from the storage buffer to the main memory. The timing of this sweeping is “the time when a load instruction does not use the main memory”. Accordingly, sweeping from the storage buffer to the main memory is possible every time except for the timing when a load instruction is accessed from stage AT to the main memory.

The storage D is swept to the main memory at the time of t_7 , and the main memory access of the load G is performed at the time of t_8 ; and the main memory access of the load H is performed at the time of t_9 . There is a load E instruction in relation to the main memory address E in the stage AT at the time of t_9 . At that time, there is a storage E instruction in relation to the main memory address E within the storage buffer. At that time, as described in the “Secondary Example” shown (Y) of FIG 3, the storage E instruction process takes priority over the process of the load E instruction until the unexecuted storage E instruction is completed in relation to the main memory from the storage buffer therein.

At that time, because there are two storage instructions in relation to the main memory address E within the storage buffer, referring to Embodiment shown FIG 6, these two storage instructions E are given priority in execution. Then, the first storage E instruction is swept to the main memory from the storage buffer at the time of t_{10} , and the second storage E instruction is swept to the main memory at the time of t_{11} . At this point, since there are no unexecuted storage E instructions within the storage buffer, the sweeping of the storage buffer is inhibited in the sweeping controller 21; the main memory access of the load E instruction is performed at the time of t_{12} ; unexecuted storage F instruction is performed in relation to the main memory within the storage buffer at the time of t_{13} ; and the storage G instruction is performed in relation to the main memory address G at the time of t_{14} .

Further, writing a storage instruction to the storage address entry registers 2 through 4 is performed by the storage buffer address writing register 1, and writing to the storage data buffer 14 is performed by the storage buffer data writing register 12 respectively, and sweeping to the main memory from the storage buffer is performed by the storage buffer read register 29.

The sweeping controller 21 understands how many unexecuted “storage instructions that must take priority over a load instruction” exist within the storage buffer, and which is indicated by raising a flag whether or not an instruction must be swept in relation to each word. The sweeping controller 21 controls so that only flagged storage instructions can be swept. Also, after completion of the storage processing, it is necessary to return the value of the storage buffer read register to the word position of a storage instruction that is unexecuted in relation to the main memory within the storage buffer, and the sweeping controller 21 controls sweeping so that storage instructions that have already been swept in relation to the main memory cannot be swept again.

A detailed description of the sweeping controller 21 is shown in FIG 7.

Normal storage processing refers to the validity storage instruction flag buffer 23. This validity storage instruction flag buffer is a flag buffer to raise flag “1” on an assigned word at the time of the storage address writing and to perform reset at the time of storage buffer sweeping, and which shows whether or not a storage instruction for each word is executed in relation to the main memory.

Further, the value for the number of words of the storage buffer is set from 0 through N respectively in the storage buffer read assignment register buffer 27. An output signal from the validity storage instruction flag buffer 23 is encoded by the encoder 24 so that the word position of unexecuted storage instructions within the storage buffer can be assigned, and the storage buffer read register 29 is set by assigning the word position from where sweeping should be done from the storage buffer in the output signal from the storage buffer read assignment register buffer 27 through the selector 28.

When a storage instruction that has not been executed in relation to the same memory address and a load instruction are conflicting as shown in the “Secondary Example” of (Y) in FIG 3 as described above, a flag is raised for a word corresponding to the priority storage instruction flag buffer 25 under the condition where the value of match detection comparators 5 through 7 that are shown in FIG 6 in the AND circuits 33_1 through 33_n are “1” (priority storage address = successive load address); at the time of a load request processing demand; and an unexecuted storage request in relation to the main memory (output of the validity storage instruction flag buffer 23). The output signal of the priority storage instruction flag buffer 25 is decoded by the encoder 26, and this encoder 26 judges which storage instruction processing is performed, and selects the output signal from the storage buffer read assignment register buffer 27 passing through the selector 28, and sets the storage buffer read register 29. The adder 30 calculates how many storage instructions there are within the storage buffer that should be given priority to process, and sets this in the register 31. Further, every time a priority storage processing is executed, the register 31 is renewed by the decrementer 32. It is a normal storage processing when the register 31 is “0”, and when register 31 is “1”, an unexecuted storage instruction within the storage buffer and a load instruction are conflicted

concerning accessing to the same memory address, and it is used for a selection signal of the selector 28.

Resetting of the validity storage instruction flag buffer 23 is performed in relation to a flag of a word position that is going to be set in the storage buffer reading register 29. Further, a word position flag that has been read from the priority storage instruction flag buffer 25 is reset with the same timing. In other words, only a word position flag of an unexecuted storage instruction in relation to the main memory is “1” within the significant storage instruction flag buffer 23 as well as the priority storage instruction flag buffer 25.

Efficacy of the Invention

The present invention described above has an advantageous effect when there is a successive load instruction in relation to the storage instruction within an unexecuted storage buffer that precedes to the same main memory address, it has the ability to eliminate delayed accessing to the main memory of a load instruction through unnecessary storage processing, by switching the storage processing, which has priority over the load processing, to the load processing priority at the time of completion of all the storage processing in relation to the same memory address within the storage buffer.

The present invention also has an advantageous effect when there is a successive load instruction in relation to a storage instruction within an unexecuted storage buffer that precedes the same main memory address, and it has the ability to eliminate delayed accessing to the main memory of a load instruction through unnecessary storage processing by executing only the storage processing in relation to the same main memory address within the storage buffer, and switching the storage processing, which has priority over the load processing, to load processing priority at the time of completion of the processing.

4. Brief Description of the Drawings

FIG 1 is a block diagram showing one Embodiment of the present invention.

FIG 2 is a Drawing showing details of a sweeping controller in FIG 1.

FIG 3 is a time chart of the normal storage processing.

FIG 4 is a time chart of the Embodiment of the present invention.

FIG 5 is a diagram showing an example of the pipeline processing.

FIG 6 is a block diagram showing another Embodiment of the present invention.

FIG 7 is a Drawing showing details of a sweeping controller in FIG 6.

FIG 8 is a time chart of another Embodiment of the present invention.

1	Storage buffer address writing register
2 ~ 4	Storage address entry register
5 ~ 7	Match detection comparator
8	Main memory write pending address register
9	Sweeping controller
10	Storage buffer read register
11	Incrementer
12	Decrementer
13	Incrementer
14	Storage data buffer
15	Main memory write pending data register
16	Incrementer
17	Flag buffer register
18	Adder
19	Register
20	Decrementer
21	Sweeping controller
22	Decoder
23	Validity storage instruction flag buffer
24	Encoder
25	Priority storage instruction flag buffer
26	Encoder
27	Storage buffer reading assignment register
28	Selector
29	Storage buffer reading register
30	Adder
31	Register
32	Decrementer
33 ₁ ~ 33 _n	AND circuit

Patent Applicant: Nihon Electric Corporation

Agent: Masaki Yamakawa (two others)

FIG 1

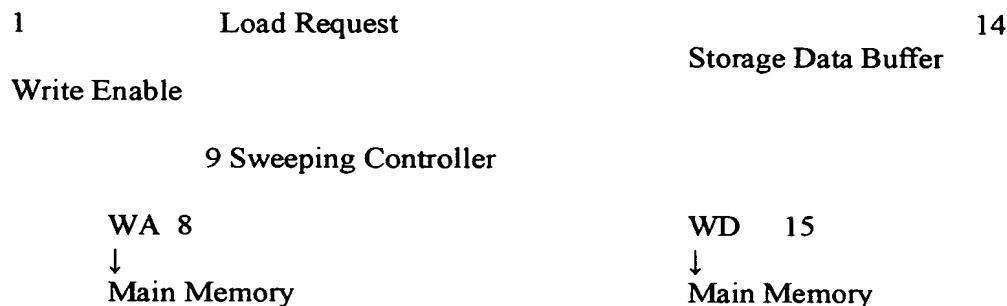
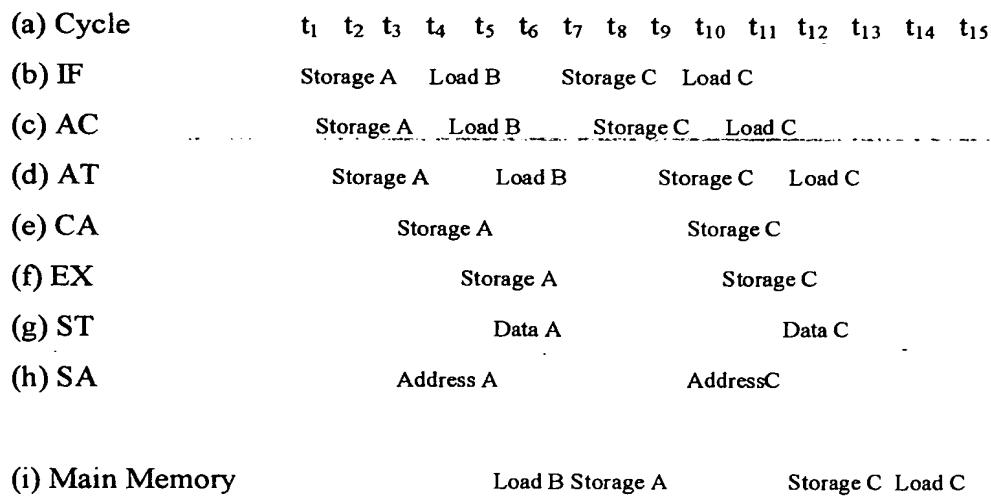


FIG 2



FIG 3



(X) (Y)

FIG 4

(a) Cycle

- (i) Main Memory**
- (j) Write Address Pointer**
- (k) Write Data Pointer**
- (l) Read pointer**

FIG 5

AIC
Instruction Cache
IR
Address Adder
LAR
TLB
PAR
Operand Cache
EXR
Operation Unit
RDR
Storage Buffer
WD
Main Memory

FIG 6

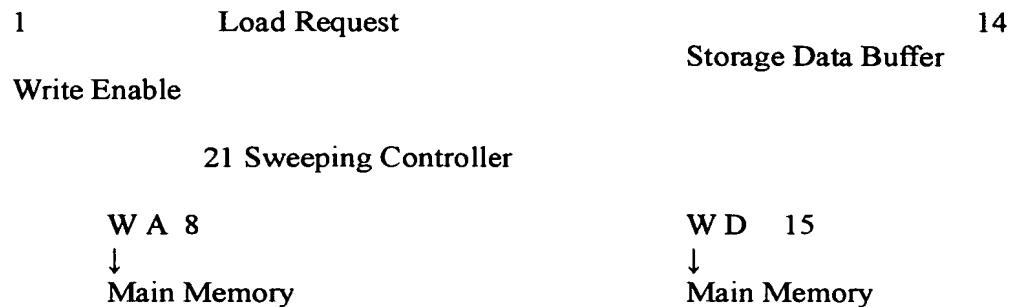


FIG 7

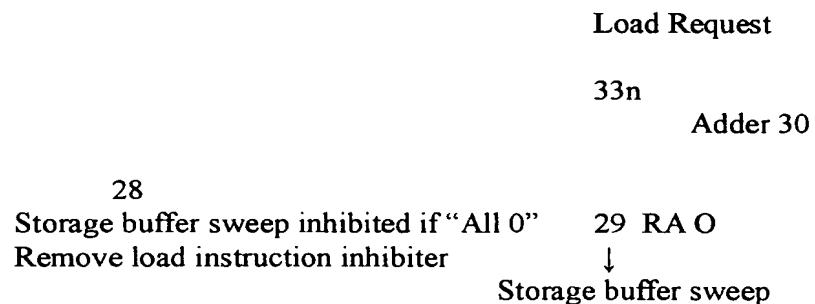


FIG 8

(a) Cycle

(i) Main Memory

(j) Write Address Pointer

(k) Write Data Pointer

(l) Read Pointer

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- BLACK BORDERS**
- IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- FADED TEXT OR DRAWING**
- BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- SKEWED/SLANTED IMAGES**
- COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- GRAY SCALE DOCUMENTS**
- LINES OR MARKS ON ORIGINAL DOCUMENT**
- REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.